All.Net Analyst Report and Newsletter

Welcome to our Analyst Report and Newsletter

Privilege Escalation and Trust

Privilege escalation has always been a problem in computers, and after all these years, we still don't seem to have a handle on the issue. I think this is closely related to our lack of understanding and systematic deployment of trust models and the nature of trust.

Non-zero trust

One of the good things about the zero trust fad is that it is bringing people awareness of the issue of trust, which cannot be zero if we are to continue to exist.

Fundamentally, trust is about the willingness to suffer harm (from others and yourself). And this goes to the issue of Risk.

Risk (the R word – the 4-letter word that ends in k)

Here I go again, trying to explain risk. I start with the fundamentals:

We (always) live in uncertain times. We cannot perfectly predict the future.

So we are always anticipating possible futures and, through our actions, seeking to constrain them. I call this process "model based situation anticipation and constraint". Since we cannot perfectly predict, we do things that may cause us harm for the rewards we might gain. Thus we trust (take risks) for rewards.

No matter what we do, the future continues to come, and even if we try at all costs to avoid all potentials for harm, the second law of thermodynamics assures us that harm will come over time. Indeed the whole concept of harm really has to do with our perception of desired and undesired futures.

Privilege as a concept

As a concept, in computers, we associate activities to authorities to place limits on those activities. Privilege is a way we express those authorities, through authorizing activities. The authorized activities at a time are often called privileges (which are selectively granted, as opposed to rights which presumably are always guaranteed). Privileges are granted to mechanisms performing activities, and activity mechanisms may be associated with various things like people or devices. These activity mechanisms and associations are modeled and other mechanisms for controlling the models used to control the activity mechanisms are granted privileges. Such mechanisms are often identified with Identity Management (IdM).

Privilege escalation and de-escalation

The concept of privilege escalation should be a misnomer, but unfortunately, it is not. Most current systems provide means for mechanisms to gain MORE available actions, called privilege escalation. And in many cases, those same mechanisms may be granted LESS available actions, called privilege de-escalation, typically when they complete use of the mechanisms requiring MORE available actions. Conceptually, with great power comes great responsibility, which is to say, trust that may be abused by 'irresponsible' people or things. Perhaps we should say 'differently responsible', because of the equities issue and loyalties.

Because of the limited imagination of those who designed most of these underlying systems long ago, and also in part because of the complexity of concepts and available resources to consider and implement them, at the operating system level, there are often “rings” of trust, with “inner” rings able to do everything outer rings can do and additional things as well. Thus as a mechanism moves from outside to inside, it is granted more privileges (things it can do), but doesn’t usually lose its previous privileges.

Mechanisms interact and change with time

The actual mechanisms are not typically granted changed privileges, but rather allowed to use mechanisms with MORE privileges, thus it can be complex to coordinate between the activities, making finer grained controls harder to implement across levels. Further, the different mechanisms are usually written by different people and teams, and thus the interfaces between the mechanisms are the only way they can communicate. And of course each may change with time, making the maintenance of the mechanisms potentially problematic and uncoordinated. Having fun yet?

Modeling mismatches

This mismatch of models between the mechanisms of trust at the lower (more trusted) levels of most operating environments and the actual requirements of getting the job done (whatever those actions are that are required for the authorized activity) produces holes in the mechanism that attackers drive trucks through.

Take for example roles and rules, one of the more common IdM models, a model for authorizing actions specified in the “rules” to actors undertaking authorized activities to get a job done (carry out activity associated with roles). The “role” of “auditor”, for example, should never be authorized to modify anything the mechanisms of the auditor are auditing. But the mechanisms used to grant that “read-only” access is a mechanisms that has more than just “read” access. The escalated privilege mechanism of granting read-only access usually has “write” access to lots of things, and it needs some form of “write” access to whatever thing is written to change the access of the mechanisms acting in the role of “auditor” from whatever previous access it had to “read-only” access related to the things under audit.

These sorts of excessive privileges are exploited by attackers. Indeed many of the underlying mechanisms grant excessive privileges and don’t de-escalate from other authorities already granted even while taking those authorized escalated actions.

Log4j and pkexec as examples

In the Linux world, where “setUID” (or setGID) or similar mechanisms are used to grant mechanisms (programs) authority to act at escalated privileges (the privileges associated with the UID (or GID) they are set to), two such mechanisms have been identified as and exploited for having weaknesses allowing their escalated privileges to be exploited by attackers in the last few months. In each case, these mechanisms were granted “root” privileges (do anything to anything), when they didn’t need any such thing for any legitimate use.

They were trusted but not worthy of that trust. But let’s get clear on this. No program is worthy of that much trust. Unlimited authority to do anything is simply never necessary. But the problem is, creating specialized programs for each specific thing requiring privilege escalation introduces more programs that can have more errors. Using common libraries or mechanisms means that any error in any of them might produce large-scale exploitation.

It is NOT a technically unsolvable problem

This basic problems of privileges and limitations, the complexity of fine grained control, the increased likelihood of errors with more code and consequences for reuse of code, the infeasibility of proving any program performs as desired or adequately specifying the desired performance of programs, all add up to a technical nightmare. But the nightmare stems from the desire for the unachievable goal of technical perfection.

It is a governance and management problem not well addressed

The inherent nature of the problem stems from a lack of governance and management surrounding the notion of trust and how to manage trust. The technology can address some set of technical issues, but it cannot address the lack of adequate trust modeling by executive management.

The governance non-zero trust problem

The governance problem comes from a lack of decision-making surrounding several key trust issues. That is, how much trust (i.e., how much loss) to place in how many entities and how to distribute it. Here are a few examples from the Standards of Practice that go to this issue:

- **Risk aggregation:** How do we control the aggregation of risk (trust/consequences)?
  - As a fundamental, the aggregation of risk (trust/consequences) in an ever smaller number of mechanisms (often the mechanisms of so-called zero trust) leveraged to increase effectiveness and reduce costs, produces an imbalance between surety (the certainty of the mechanism) and consequences/trust we place in them. And isn't that same trust used to achieve desired futures? What are the decisions about limitations on this increased efficiency versus the effects of a failure in the highly efficient mechanism? How has governance addressed this?
- **Separation of duties:** How do we separate duties to prevent what sized groups of what/whom from collaborating to what effect?
  - How do we limit our trust/consequences in individuals and groups of people and mechanisms so that groups of different sizes colluding to cause harm are limited in the harm they can produce? Note that we also must place trust in them for the positive futures they may bring. Obviously, a single group or system should not be able to create harm beyond some threshold, but analysis of the effects of groups is more complex because of synergistic effects, such as the ability to create a falsehood and inducing a failure to detect it. And are we also then limiting that individual or group from generating benefits? How do we limit harm but not benefit?
- **Change control:** How do we manage changes to limit potential harm?
  - As systems and people change, those changes each produce potential effects on consequences. Thus how much trust do we place in changes and the mechanisms of change? How do we separate duties and mitigate trust aggregation in changes? Changes happen at multiple scales and time frames. How do we allow emergency changes to prevent undesired futures at critical times without allowing changes that cause undesired futures at critical times? When we terminate employment, how do we control the changes in privileges associated with the old and new worker with the associated responsibilities for producing the desired futures?

- **Common mode failures:** How do we mitigate common mode failures?
  - Most of the widest ranging consequences stem from too many mechanisms placing trust in too few dependency chain members. The death of 1,000 cuts by the same knife. How much info-diversity and cyber-diversity do we want and how much trust to we put in each part at what volume? How much dependency are we already placing in what few parts? This is an inventory problem as well as a supply chain and demand-chain problem, and just-in-time efficiencies bring substantial rewards that tend to be reduced when we require elimination of common-modes. Efficiency trades with effectiveness, and anticipating the range of futures, how do we place our bets?
- **Complexity and cost:** How do we make it simple enough to implement reliably?
  - All systems solutions must be simple enough to actually work. The more complex the solution, the harder it is to do at all, and complexity drives increased costs and error rates. Just starting to address all of the issues identified here seems beyond the desire for complexity and cost for most enterprises. While tools can help us reduce this, we then transfer the trust to the tools and those who use them. How do we decide how much to trust to put in the theories behind the mechanisms we use to analyze trust?

Transitive and time-transitive trust and escalation

When I trust you for something, I am also trusting everyone and everything you trusted, trust, or may trust, in the path to that thing. Thus trust is transitive. Furthermore, it is time transitive in that something you trusted at some time in the past may effect actions you take now or in the future. Escalation then implies time transitivity of escalation trusts. So escalation for a short period of time may leave lingering effects.

Everyone in cyber-security knows this already because they understand that a privilege escalation can plant a Trojan horse for subsequent reuse. That's how remote access Trojans (RATs) work most of the time. But as a field, this has been widely ignored in the more subtle aspects of how you deal with a breech. Of course in the computer virus field, this was published in the 1980s, and before that I'm sure in related areas. The time transitivity of information flow is well understood at a theoretical level, and the time transitivity of trust is very similar.

From a risk/trust standpoint, we cannot realistically de-trust everything as soon as we detect something anticipatory of an undesired future. We could try to track back-track all trust escalations and search for all potential consequences of those escalations, but I strongly suspect that the analysis grows exponentially or worse (the real exponential, not the catch phrase people use to mean somehow bad). And the traces provided in modern systems of activities are not fine enough granularity to do this precisely, thus we would either have to assume (trust) that certain bad things did not happen or assume that some bad things that could happen did happen, and do so at some less than ideal level of granularity.

Conclusions

This is just the start of this discussion, but I think we now all understand that this zero trust thing is problematic (and non-existent) and that privileges and escalation are about trust and that trust is about governance decisions, usually not yet made. It's a start...

